Programação Visual
Trabalho de Laboratório nº 11

Objetivo	Angular – Introdução à utilização de Angular em projetos ASP.NET Core MVC. Criação de componentes, serviços, binding e *pipes*.
Programa	Pretende-se continuar o desenvolvimento da aplicação ESTeSoccer.
Regras	Implementar o código necessário e testar no fim de cada nível. Use as convenções de codificação adotadas para as linguagens usadas.

Nível 1

- Descarregue e descompacte o ficheiro "**Lab 11 – Materiais**" fornecido com este enunciado.
- Inicia uma instância do serviço (ESTeSoccer) e outra do client Angular (ESTeSoccerAngularClient) e garanta que ambos estão a correr corretamente e que o client de Angular está a receber os dados das ligas.
  Nota: Caso tenho algum problema relativo aos dados, garanta que tem a base de dados do lab anterior, ou então apague, e corra novamente os comandos de Add Migration e Update Database.

  Anteriormente (lab 10) utilizamos interfaces para permite que o Angular soubesse que dados estamos a trabalhar, vamos agora criar uma estrutura adequada o que nos irá permitir utilizador outro tipo de funcionalidades.

- Crie uma class com os campos LeagueId, Name, e Country, e um construtor para inicializar estes campos, na pasta src/app através do comando:
  `ng generate class league`
- Crie um serviço através do comando:
  `ng generate service league`

- Crie um metodo "getLeagues" que irá chamar o nosso controller (.net) à semelhança do que já existe no componente leagues.
- Substitua o código do construtor do componente Leagues, pela injeção deste novo serviço de modo a continuar a obter as ligas. Não se esqueça de eliminar a interface e utilizar a nova classe League.
- Teste a aplicação Angular

Nível 2

- Adicione um metodo "deleteLeague" no LeagueService e faça as alterações necessárias para utilizar este método a partir do LeagueComponent.
- Adicione um método "getLeague" no LeagueService, deve receber um id e retornar a liga com o respetivo id.
- Altere o componente LeagueDetailsEdit, de forma a usar o novo serviço.
- Adicione um método "editLeague" no LeagueService, deve receber um objeto to tipo League e chamar o endpoint correspondente para editar uma liga.

Programação Visual
Trabalho de Laboratório nº 11

- Altere o component LeagueEditComponent para utilizar o novo serviço em vez de chamar diretamente a API.
- Caso tenha um erro ao tentar editar uma liga, lembre-se de que o uso de Observable é asincrono, desta forma o construtor termina antes do pedido retornar, e ao iniciar o template (pagina) não existe ainda nenhum liga carregada. Para resolver este problema deve utilizar um mecanismo semelhante ao que existe no LeaguesComponent, onde se utiliza uma diretiva para verificar se a variable league já está definida. Adicione a diretiva `*ngIf="league"` a sua div inicial do componente.
- Teste a aplicação.

Nível 3

- O Angular oferece *data binding*, que é o que permite que o frontend aguarde que algo no seu "backend" termine de correr e seja atualizado posteriormente. Desta forma, ao alterar dados nos components ou serviço (model), obtemos imediatamente uma atualização no frontend (view), isto é, o model está binded à view.
- Altere o html do LeagueEditComponent, substituindo a linha com o elemento h1, por esta:
```
<h1 id="tableLabel">League {{league.leagueId}} - {{league.name}}
</h1>
```
- Teste a aplicação. Irá reparar que ao editar uma liga e trocar o nome, quando clica "Update" o nome que aparece no h1 também é alterado.

  Um dos tipos de data binding que o Angular oferece é o "*Two Way Data Binding*", o que significa que tanto o model está binded a view como a view está binded ao model, isto permite que os dados sejam alterados no model ao alterar na view, como por exemplo um input.

Nível 4

- Acrescente a seguinte diretiva no input para o nome da liga no LeagueEditComponent:
```
<input id="name" [(ngModel)]="league.name"  type="text"
formControlName="name">
```
- Faça o mesmo teste de antes, repare que agora não precisa de clicar em "Update" para os dados mudarem no frontend.
  Nota: Sendo estes dados geridos por um backend (.net) temos de clicar em "Update" para que estes dados sejam realmente guardados.
- O Angular oferece uma forma bastante simples de fazer algumas transformações de dados através de Data Pipes. Altere novamente a linha com o H1 de forma a utilizar o UpperCasePipe no nome da liga.

Programação Visual
Trabalho de Laboratório nº 11

Nível 5

- O serviço que criamos limita-se a enviar pedidos http e a retornar a sua resposta, mas não está preparado para nenhum tipo de erro. Adicione o seguinte método na classe LeagueService e importe o `HttpErrorResponse` e o `throwError`:

```
private handleError(error: HttpErrorResponse) {
  if (error.status === 0) {
    // A client-side or network error occurred. Handle it accordingly.
    console.error('An error occurred:', error.error);
  } else {
    // The backend returned an unsuccessful response code.
    // The response body may contain clues as to what went wrong.
    console.error(
      `Backend returned code ${error.status}, body was: `, error.error);
  }
  // Return an observable with a user-facing error message.
  return throwError(() => new Error('Something bad happened.'));
}
```

- Podemos agora usar o operador catchError, juntamente com as funções pipe da biblioteca RxJs, de forma a generalizar o nosso *error handling*.
- Altere os métodos do LeagueService, de forma a que a chamada ao http seja encadeada com um pipe que usa o operador catchError para chamar o nosso método handleError.

```
this.http.[method]<[model]>([url]).pipe(catchError(this.handleError));
```

- Teste esta alteração tentando aceder aos detalhes de uma liga que não existe através do Id passado como parametro no URL. Deverá aparecer um erro na sua consola do browser.

Desafio

- Apague os componentes counter e fetch-data.
- Na pasta do Angular, corra o comando ng test. Irá obter uma interface para os tests do Angular. Vão existir alguns erros. Terá de corrigir alguns setups dos testes relativamente à injeção de depedências.
- Pode começar por ver este post: https://stackoverflow.com/questions/47236963/no-provider-for-httpclient
- Vá procurando informação sobre os errors e corriga os problemas até obter todos os testes válidos.

Notas

Para os identificadores do C# siga as convenções adotadas, nomeadamente:

- A notação camelCase para o nome das variáveis locais e identificadores privados.
- A notação PascalCase para os nomes públicos dos métodos e classes
- Não utilize o símbolo de qualquer forma '_' nos identificadores
- Não use abreviaturas